6

Query Answering in Information Systems with Integrity Constraints

François Bry
Institut für Informatik, Ludwig-Maximilians-Universität München
Oettingenstraße 67, D - 80538 München, Germany
phone: 49 89 - 2178 2210 - fax: 49 89 - 2178 2211
bry@informatik.uni-muenchen.de

Abstract

The specifications of most of the nowadays ubiquitous informations systems include integrity constraints, i.e. conditions rejecting so-called "invalid" or "inconsistent" data. Information system consistency and query answering have been formalized referring to classical logic implicitly assuming that query answering only makes sense with consistent information systems. In practice, however, inconsistent as well as consistent information systems need to be queried.

In this paper, it is first argued that classical logic is inappropriate for a formalization of information systems because of its *global* notion of inconsistency. It is claimed that information systems inconsistency should be understood as a *local* notion. Then, it is shown that minimal logic, a constructivistic weakening of classical logic which precludes refutation proofs, provides for local inconsistencies that conveniently reflect a practitioner's intuition. Further, minimal logic is shown to be a convenient foundation for query answering in information systems with integrity constraints. Finally, an approach to answering queries in consistent as well as inconsistent information systems is proposed.

Keywords

Query answering, integrity constraints, database and information system integrity, paraconsistent reasoning, minimal logic.

1 INTRODUCTION

Information systems can be roughly defined as versatile data repositories that are organized independently from the computing tasks they are used for. Since declarative specifications are convenient means to ensure the desired task independence, logic-based formalizations of information systems and related issues such as query answering have been proposed and used in preference to

Integrity and Internal Control in Information Systems Volume 1
S. Jajodia, W. List, G. McGregor & L. Strous (Eds) © 1997 IFIP. Published by Chapman & Hall

other approaches. Among other things, this "logic approach" has led to the relational and deductive data models, to the formalization in logic of integrity constraints, and to integrity verification methods e.g. (Nicolas 1982, Decker 1986, Lloyd *et al.* 1987, Bry *et al.* 1988).

In this paper, while keeping with the logic approach, it is argued that the traditional reference to classical logic is not appropriate to formalize information systems. Instead, it is submitted that a weakening of classical logic, namely minimal logic, more adequately conveys a practitioner's intuition of query answering in information systems with integrity constraints.

Informally, integrity constraints are application dependent conditions rejecting certain data combinations as "invalid" or "inconsistent". A rather simple integrity constraint can for example require some data representing ages to be nonnegative or within a certain range. Another, more complex integrity constraint can relate some data items to others, requiring for example for every data item in a certain set, certain data items to occur in some other set: in this way can be ensured that for every course given in a school at a certain time, a room is available at this time. As theses examples suggest, integrity constraints to be found in information systems range from elementary, easy to check properties, to intricate interrelationships between data sets, whose verification and maintenance may be not obvious.

It is usual to distinguish as follows between "static" and "dynamic" integrity constraints. While static integrity constraints express conditions on each state of an information system, dynamic integrity constraints express conditions for changes from one state of an information system to another to be acceptable. The integrity constraints in the examples given above are all static. A common example of a dynamic integrity constraint is a condition requiring salaries never to be decreased. In this paper, only static integrity constraints are considered. In the following, "integrity constraint" is always to be understood as "static integrity constraint".

As far as the theory is concerned, integrity constraints can be reduced to queries, integrity verification to query answering. Indeed, the condition defined by an integrity constraint can be expressed as a query the answer to which is "yes" or "no", depending whether the condition is satisfied or not. For efficiency reasons, however, it is preferable not to verify integrity constraints like queries are evaluated. Instead, so-called integrity verification methods (e.g. (Nicolas 1982, Decker 1986, Lloyd *et al.* 1987, Bry *et al.* 1988)) have been developed, that, taking benefit from the satisfaction of the integrity constraints prior to an update, incrementally evaluate them in the updated information system. Building upon the above mentioned theoretical reduction of integrity constraints to queries, the standard formalization of integrity constraints, like that of query answering (Green 1969, Gallaire *et al.* 1984), is in terms of proofs in classical logic.

In this paper, it is claimed that this standard formalization is not satisfying. While it is impeccable if the integrity constraints are satisfied, it leads to an

unacceptable formalization of query answering if some integrity constraints are violated. In practice, as it is discussed in the next section, inconsistent as well as consistent information systems need to be queried. Thus a sensible formalization of query answering in inconsistent information systems is needed.

The inadequacy of classical logical as a basis for a formalization of information systems is shown to come from its *global* notion of inconsistency. It is claimed in this paper that information systems inconsistency should be seen as a *local* notion. Minimal logic, a constructivistic weakening of classical logic which precludes refutation proofs, is shown to provide for the very notion of local inconsistencies which reflects the common intuition of inconsistencies in information systems. Minimal logic is further shown to give a convenient foundation for query answering in information systems with integrity constraints. In the companion paper (Bry 1996), minimal logic was shown, thanks to its local notion of inconsistencies, to be convenient for a formalization of (nonmonotonic) negation in logic programming and deductive databases.

Finally, an approach to answering queries in consistent as well as inconsistent information systems is proposed. This approach basically consists in "labeling" answers computed from inconsistent subparts of an information system. Thus, two kinds of answers can be distinguished: those that result from data not violating any integrity constraint, which will be called "consistent answers", and those computing from data violating some integrity constraints, which will be called "inconsistent answers". It is argued that this notion of answer naturally conveys a common intuition and practice of information systems.

Information system integrity issues are of a considerable practical relevance. In spite of some research results, integrity issues have not yet received from the industrial developers of information and database systems the attention it deserves. As a consequence, information system practitioners still mostly rely on ad hoc application programs that, unfortunately, often are prone to conceptual errors. The author of this paper believes that answering queries posed to information systems that may violate some of their integrity constraints is an important issue which deserves more attention. To the best of his knowledge, this issue has not been previously investigated.

This paper consists of 7 Sections. Section 1 is this introduction. In Section 2 the issue investigated in the paper, namely answering queries in information systems in which integrity constraints may be violated, is motivated by practical considerations. It is explained why the notion of inconsistency implicitly considered by information system practitioners should be formalized as a local notion. Section 3 is devoted to a representation of information systems in logic. In Section 4, minimal logic is introduced and compared with classical logic. Minimal logic is shown in Section 5 to provide with the very notion of local inconsistency reflecting the common intuition of information systems. In Section 6, the approach to answering queries in consistent as well as inconsi-

stent information systems is proposed. Section 7 contains concluding remarks and draws perspectives for further research.

2 MOTIVATION

The purpose of this section is to set the stage. First, the view of information systems as consisting of constructive and normative specifications is recalled. Then, it is argued that, in practice, inconsistent information systems might well be queried. Finally, the notion of inconsistency referred to in information systems is claimed to be local.

2.1 Constructive and Normative Specifications

In this paper, we shall adopt the view of an information system consisting of specifications of two kinds, constructive and normative specifications.

Let first illustrate this view on example. Information systems providing with airline time tables often compel to this partition. A constructive part of the information system gives the available flights, while additional, normative specifications mention restrictions such as the latest landing times (e.g. no landings in Zürich can be scheduled after 10:20 p.m.) or the minimal connection times of an airport (e.g. in Zürich no connections with less than 40 minutes should be booked because of luggage transit).

Note that most information systems nowadays in use do not fully specify the normative specifications they are based upon. Often, they leave some normative specifications implicit, what in general complicates their maintenance. In the following, only those normative specifications are considered that are explicitly specified. In other words, misfunctions due to an incorrect modelling are not further considered.

The constructive specifications can be given in various formalisms, e.g. as the relations of a relational database, the tuples of which are explicitly stored, or as the relations of a deductive database, the tuples of which are either explicitly stored or intentionally defined from stored tuples by deduction rules, or as an object-oriented database, whose inheritance hierarchy provides with intentional specifications comparable with that of a deductive database, or in any other manner. How the constructive part of the information system is specified influences the techniques used for answering queries. However, this is not relevant for the purpose of the present paper, for which it suffices to assume that a query answering method is available.

The normative specifications are expressed as integrity constraints, if the management system support integrity constraints, otherwise as application programs that are run at every changes of the constructive specifications. For the purpose of the present paper, how the normative specifications are maintained is not relevant. It suffices to assume the presence of normative spe-

cifications. In the rest of the paper, the denomination "integrity constraints" refers to the normative specifications, regardless of whether they actually are processed as such by the system, or, less conveniently, as application programs.

Possibly, not only the constructive but also the normative specifications are used for answering queries. The fact that, say, no Air France flights from Paris to Zürich on Monday is scheduled to land after 11 p.m. can be computed either from the flights registered in the information system, or from the above mentioned prohibition of late (scheduled) landings in Zürich. Researchers have investigated so-called "semantic query evaluation" methods e.g. (Chakravarthy *et al.* 1990, Cholvy 1990, Gal and Minker 1988) that, if possible, make benefit of (satisfied) integrity constraints for answering queries like this one. Although yet not used in marketed database systems, these techniques are promising. The framework of the present paper does not preclude them.

In the following, an information system will be said to be "consistent" if its constructive specifications fulfil its normative specifications. Otherwise, it will be said to be "inconsistent". Considering once again the above mentioned airline time table information system and assuming that it has no other integrity constraint than the one precluding scheduled landings in Zürich after 10:20 p.m., this information system is consistent if no flights are – extensionally or intentionally – defined with landings in Zürich after 10:20 p.m. In Section 3, the definition of consistent information systems is formalized in logic.

2.2 Querying Inconsistent Information Systems

Theoreticians often implicitly or explicitly assume that only consistent information systems are queried. In practice, however, inconsistent information systems have to be dealt with. There are several reasons for this.

First, already during the design phase, before the consistency of the information system is established, queries need to be answered. This is obviously the case of complex information systems, like airline time tables, that are difficult to design, whose design usually lasts over long periods of time, and for which taking late design decisions often require querying the previous, provisional design.

Second, consistency sometimes is restored by another agent than the one who caused of the inconsistency in the first place. In the time between the update responsible of the inconsistency and the consistency restoring action, the information system often still stays in use. This might for example be the case of a banking application in principle forbidding certain updates, in practice nevertheless accepting certain transactions, that strictly speaking should not be allowed (e.g. a bank might accept overdrafts over a permitted amount from reliable customers).

Finally, restoring consistency often requires to query the (inconsistent!) information system. Indeed, the integrity restoring actions often depend on

the state of the affairs, which sometimes cannot be known without querying the inconsistent information system.

2.3 Locality of Information System Inconsistencies

Anticipating the next section, let be recalled that answers to queries posed to an information system *IS* are defined as logical consequences of *IS*, the information system *IS* itself been seen as a logical specification.

The notion of logical consequence is classically defined in logic in model theoretic terms. Basically, a model is a (mathematical) object fulfilling the considered specification. Relying on this notion of model, an expression A is an answer to a query posed to an information system *IS* if A is true in every possible model of *IS*.

This informal definition calls for two remarks.

First, an answer A is related to a query Q by the condition that A is an instance of Q. This point is not relevant here.

Second, this definition of an answer is not used for computing answers, for it is of course not possible to consider all possible models of a specification. Instead, query answering procedures are used that are based on proof methods. The considered proof methods are proved to reflect the model theoretic definition of logical consequences.

If some integrity constraints are violated in an information system, the information system can be viewed as a (logically) inconsistent specification. Basically, a logical specification is inconsistent if it contains contradicting requirements, like e.g. a formula F and its negation $\neg F$. Clearly, a (logically) inconsistent specification has no models, for contradicting specifications cannot be satisfied. As a consequence, every possible expression A is an answer in an inconsistent information system *IS*: A is trivially true in every model of *IS* as soon as *IS* has no models.

This view of an answer in an inconsistent information system does not reflect the intuition. If for example a personal management database imposes ages to be not less than 18 years and neither extensionally nor intentionally specifies Jack's salary, no one expects the database to answer that this salary is 500.000 $ just because Mary's records mention an age of 15 years! Within classical logic, however, this answer is perfectly correct. Indeed, in classical logic every formula can be derived from an inconsistent specification.

Intuitively, it is acceptable that illegal data locally corrupt *some* answers, but not that they affect *all* possible answers. In the previous example, if Jack's salary is not related to Mary's age, one expects the incorrect recorded age for Mary not to affect Jack's salary in a sensible implementation of an information system. Thus, from a practical viewpoint it does not make sense that every possible expression is a consequence of an inconsistent information system.

3 A LOGIC VIEW OF INFORMATION SYSTEMS

In this section, a simple formalization of information system in logic is recalled. This formalization is shown to encompass relational and deductive databases of different types. Familiarity with the basic notions of logic (as introduced in e.g. (Fitting 1996)) is assumed.

A logic language with the following logical symbols is assumed: $\forall$, $\exists$, $\wedge$, $\vee$, $\rightarrow$, $\neg$, and $\bot$, an atom always false. The negation $\neg F$ of a formula F is defined as a shorthand for $F \rightarrow \bot$.

In the considered framework, an information system is simply formalized as a finite set of closed first-order formulas, the negated formulas expressing the integrity constraints or normative specifications, the unnegated formulas, the constructive specifications. Closedness of a formula F means that every variable occurring in F is quantified in F. Informally, the restriction to first-order logic means that variables range over individuals of the universe of discourse. Recalling that $\neg F$ is viewed as a shorthand for $F \rightarrow \bot$, where $\bot$ denotes the formula false in every interpretation, it is quite intuitive to view negated formulas as integrity constraints: $F \rightarrow \bot$ forbids states in which F is true, i.e. constrains the information system to states in which F is false.

The restriction to first-order formulas, although common, is questionable because higher-order constructs are needed, if so-called "aggregates" such as average values, cardinalities of sets, and maximum or minimum values are to be expressed. However, such higher-order constructs usually occur in queries and possibly in the normative part of an information system, not in its constructive part, thus making the proposed formalization of information system acceptable. Indeed, in most applications it makes more sense to have a normative specification requiring a total cost to be no more than 200 $ than to have a constructive specification *defining* this total cost.

This formalization of information systems is purely syntactical, for it does not presume of the formulas interpretation.

Imposing further syntactical restrictions yields well known database models.

3.1 Relational Databases

The constructive specifications of a relational database are ground atoms, called tuples. (A relation consists of all ground atoms with same predicate symbol.) Integrity constraints are negated closed formulas.

3.2 Definite Deductive Databases

In a definite deductive database constructive specifications are ground atoms, called facts, or rules of the form:

$$L_1 \wedge ... \wedge L_n \rightarrow A$$

where the L_i are literals, $n \geq 1$, and A is an atom. Such a rule implicitly stands for its universal closure. Integrity constraints are negated closed formulas of any kind, like in relational databases.

3.3 Disjunctive Deductive Databases

The constructive specifications of a disjunctive deductive database (Lobo *et al.* 1992) are ground atoms, or disjunctions of ground atoms, or rules of the form:

$$L_1 \wedge ... \wedge L_n \rightarrow A_1 \vee ... \vee A_m$$

where the L_i are literals, $n \geq 1$, the A_j are atoms, and $m \geq 1$. Such a rule implicitly stands for its universal closure. Integrity constraints are negated closed formulas of any kind, like in relational and definite deductive databases.

In disjunctive deductive databases, integrity constraints have not been much investigated. Because of the indefiniteness of disjunctive deductive databases, it is not clear, how to define integrity constraint satisfaction in disjunctive deductive databases: Should an integrity constraint refer to one model of the disjunctive database, or to all its models? Consider for example a disjunctive database D with constructive part $F = \{$ *colour(garfield, white)* $\vee$ *colour(garfield, black)* $\}$, and with an integrity constraint $C =$ *colour(garfield, white)*. Should C be seen as satisfied in D? It is often considered that D does not satisfy C, for there are models of D in which C is falsified.

Note, however, that how integrity constraint satisfaction is defined in disjunctive deductive databases does not affect the issue addressed in the present paper.

Disjunctive deductive databases generalize definite deductive databases, and definite deductive generalize relational databases. Inheritance hierarchies, as occurring in object-oriented databases, can be expressed by means of rules. Thus, even though possibly less conveniently, deductive databases can be seen as a generalization of object-oriented databases.*

In the following, a – definite or disjunctive – database will be called "positive" if none of its rules contain negative literal in its premise.

*However, in contrast to object-oriented databases, deductive databases preclude cyclic terms.

4 MINIMAL LOGIC

In this section minimal logic is recalled. Minimal logic is a constructivistic, or intuitionistic weakening of classical logic. It is defined in terms of a proof system, called natural deduction, introduced by Gentzen in (Gentzen 1934), which is first recalled after (Prawitz 1965).

4.1 Natural Deduction Briefly Recalled

A *deduction* $\mathcal{D}^F$ of a formula F in a theory IS is a tree, the nodes of which are occurrences of formulas. The root of $\mathcal{D}^F$ is an occurrence of F. The leaves of $\mathcal{D}^F$ are occurrences of formulas in IS, or are *open assumptions.* Open assumptions are formula occurrences that can, later in the proof, be *discharged* by applications of certain inference rules. Formula occurrences are distinguished from formulas, because an application of an inference rule discharges one formula occurrence, but not all open assumptions in the already expanded deduction that are occurrences of the same formula.

Deductions $\mathcal{D}^F$ of formulas F in a theory IS are inductively defined together with the set $O(\mathcal{D}^F)$ of their open assumptions as follows:

1. If $F \notin IS$, a one node tree $\mathcal{D}^F$ consisting of an occurrence of F is a deduction of F in IS, the only open assumption of which is the occurrence $\mathcal{D}^F$ of F itself.

2. If $F \in IS$, a one node tree $\mathcal{D}^F$ consisting of an occurrence of F is a deduction of F in IS with no open assumptions.

3. For $k = 1, ..., n$ let $\mathcal{D}^{F_k}$ be a deduction of F_k in IS with set of open assumptions $O(\mathcal{D}^{F_k})$.

   $\mathcal{D}^F :=$ $\dfrac{\mathcal{D}^{F_1} \ \ldots \ \mathcal{D}^{F_n}}{F}$ is a deduction of F in IS if $\dfrac{F_1 \ \ldots \ F_n}{F}$ is

   an application of one of the inference rules below, the open assumptions of which are the formula occurrences in the sets $O(\mathcal{D}^{F_k})$ $(k = 1, ..., n)$ except, possibly, for discharged formula occurrences. A discharged occurrence of a formula F is indicated in the inference rules below by (F).

Introduction rules:

$\wedge$I $\quad \dfrac{A \quad B}{A \wedge B}$

$\vee$I$_r$ $\quad \dfrac{A}{A \vee B}$

$\vee$I$_l$ $\quad \dfrac{B}{A \vee B}$

$\rightarrow$I $\quad \dfrac{\begin{matrix}(A)\\ B\end{matrix}}{A \rightarrow B}$

$\forall$I $\quad \dfrac{A}{\forall y A[y/x]}$

$\exists$I $\quad \dfrac{A[t/x]}{\exists x A}$

Elimination rules:

$\vee$E $\quad \dfrac{A \vee B \quad \begin{matrix}(A)\\ C\end{matrix} \quad \begin{matrix}(B)\\ C\end{matrix}}{C}$

$\wedge$E$_r$ $\quad \dfrac{A \wedge B}{A}$

$\wedge$E$_l$ $\quad \dfrac{A \wedge B}{B}$

$\rightarrow$E $\quad \dfrac{A \quad A \rightarrow B}{B}$

$\forall$E $\quad \dfrac{\forall x A}{A[t/x]}$

$\exists$E $\quad \dfrac{\exists y A[y/x] \quad \begin{matrix}(A)\\ B\end{matrix}}{B}$

Absurdity rule:

$\perp_c$ $\quad \dfrac{\begin{matrix}(\neg A)\\ \perp\end{matrix}}{A}$

Condition on the rule $\forall$I: y must not occur free in any open assumption on which A depends, i.e. y must not occur free in any formula occurrence in $\mathcal{O}^A \cup O(\mathcal{D}^B)$.

Condition on the rule $\exists$E: y must not occur free in $\exists x A$, or in B, or in any assumption on which the upper occurrence of B depends other than A, i,e. y must not occur free in any formula occurrence in $\{\exists x A, B\} \cup O(\mathcal{D}^B)$.

If $\mathcal{D}^F$ is a deduction of a formula F in a theory IS with no open assumptions, i.e. if $O(\mathcal{D}^F) = \emptyset$, then it is a *proof* of F in IS, the existence of which is, as usual, denoted by $IS \vdash F$.

4.2 Minimal and Classical Logic Compared

Classical and minimal logic are simple to define and compare in terms of natural deduction: For deductions in classical logic, all the inference rules

can be applied. For deductions in minimal logic, only the introduction and elimination rules can be applied, applications of the absurdity rule $\perp_c$ are precluded.

Let $\vdash_c$ and $\vdash_m$ denote provability in classical and minimal logic, respectively. Classical logic provability can also be defined by replacing the absurdity rule $\perp_c$ with the elimination of double negation rule $\dfrac{\neg\neg A}{A}$ or, alternatively, as minimal logic provability from a (infinite) theory consisting of so-called stability axioms $\neg\neg A \rightarrow A$ for every formula A, as shown by the following classical logic deduction (the numbers indicates the rule application by which a formula occurrence is discharged, recall that $\neg A$ stands for $A \rightarrow \perp$):

$$\dfrac{\dfrac{(2)\ \neg A \qquad\qquad (1)\ \neg\neg A}{\perp}\ \rightarrow \mathrm{E}}{\dfrac{A}{\neg\neg A \rightarrow A}\ \perp_c (2)}\ \rightarrow \mathrm{I}\ (1)$$

4.3 Classical Logic Global Notion of Inconsistency

The absurdity rule $\perp_c$ "globalizes" inconsistencies, in the sense that it makes it possible to derive every formula A in an inconsistent theory ("ex falso quodlibet" principle), i.e. $\vdash_c \perp \rightarrow A$, as follows from the following deduction of $\{\neg\neg A \rightarrow A\} \vdash_m \perp \rightarrow A$:

$$\dfrac{\dfrac{\dfrac{(2)\ \neg A}{(1)\ \perp}}{\neg\neg A}\ \rightarrow \mathrm{I}\ (2) \qquad\qquad \neg\neg A \rightarrow A}{\dfrac{A}{\perp \rightarrow A}\ \rightarrow \mathrm{I}\ (1)}\ \rightarrow \mathrm{E}$$

4.4 Minimal Logic Local Notion of Inconsistency

Since minimal logic does not treat the formula $\perp$ differently from any other atom, if A is any formula, $\not\vdash_m \perp \rightarrow A$. Otherwise, for every atom B distinct form $\perp$, $\vdash_m B \rightarrow A$ would hold. Let A and B be two atoms distinct from $\perp$. Since there are interpretations that do not satisfy $B \rightarrow A$, this would contradict the correctness of the inference rules.

Thus, the "ex falso quodlibet" principle does not hold in minimal logic. One

can see inconsistencies in minimal logic as "local" in the sense that they do not give rise to the derivation of every possible formula.

5 MINIMAL LOGIC AS A BASIS FOR QUERY ANSWERING

In this section, it is argued that minimal logic is convenient for a formalization of query answering in information systems. The argumentation it twofold. First, the restriction to minimal logic proofs is shown to convey a common intuition of query answering in information systems. Second, minimal logic is recalled to be sufficient a foundation for a large class of information systems.

5.1 Intuitive Argument

Informally, what is lost when minimal logic is considered in lieu of classical logic are:

1. Proofs based on an elimination of double negations, i.e. proofs of A from T resulting from a derivation of $\neg\neg A$ from T, and

2. refutation proofs, i.e. proofs of A from T resulting from a derivation of $\bot$ from $T \cup \{A\}$.

If negated formulas $A \rightarrow \bot$ are to be seen as normative specifications, i.e. as integrity constraints, as proposed in Section 3, then the elimination of double negation should not be necessary for answering queries. Indeed, if A holds in an information system IS, this means that A follows from the constructive part of IS, regardless of integrity constraints. The expression $\neg\neg A = (\neg A) \dot{\rightarrow} \bot$ corresponds to an integrity constraint meaning that $\neg A$ should not hold in the information system.

Intuitively, refutation proofs can be rejected by a similar informal argument. A proof of $\bot$ from an information system IS augmented with an hypothesis A can be seen as meaning that the extended information system $IS \cup \{A\}$ violates some integrity constraint. Since query answering is specified – if not performed – referring only to the constructive part of the information system, the violation of an integrity constraint should not give rise to compute answers that, otherwise, are not derivable.

It is worth stressing that the "argument" given in this section is questionable, for it is informal. It might however help in sustaining the intuition.

5.2 Formal Argument

Recall that a positive – definite or disjunctive – deductive database is such that no negative literal $\neg B$ occurs in the premise $L_1 \wedge ... \wedge L_n$ of some its rules $L_1 \wedge ... \wedge L_n \rightarrow A_1 \vee ... \vee A_m$. Positive deductive databases are especially interesting because how negation is to be interpreted is not obvious.

While it is commonly accepted that a nonmonotonic form of negation corresponding to "negation as failure" (Clark 1978) and to the "closed world assumption" (Reiter 1978) is needed, how negation is to be defined remains a debated question. Basically, nonmonotonic negation corresponds to the kind of common sense reasoning applied when someone concludes from a time table that there are no flights at a given time if none are mentioned in the time table. In this paper, how negation is to be interpreted in information systems is not further addressed. Instead, positive deductive databases are considered in more detail.

While minimal logic is not complete for full classical logic, it is complete for positive definite or disjunctive deductive databases without integrity constraints.

(a) Incompleteness of Minimal Logic

The incompleteness of minimal logic for full classical first-order logic is easy to establish: If $\mathcal{I}$ is an interpretation and A an atom distinct from $\bot$, then $\mathcal{I} \models \neg\neg A \rightarrow A$, as it is established in Section 4.2. Since minimal logic treats $\bot$ like any atom B distinct from $\bot$, if $\vdash_m \neg\neg A \rightarrow A$ (where $\vdash_m$ denotes derivability in minimal logic), then $\vdash_m ((A \rightarrow B) \rightarrow B) \rightarrow A$. This is incorrect, because there are interpretations $\mathcal{I}$ such that $\mathcal{I} \not\models A$ and $\mathcal{I} \models B$, i.e. $\mathcal{I} \not\models ((A \rightarrow B) \rightarrow B) \rightarrow A)$. The result follows from the correctness of the inference rules of the system of natural deduction, which is easy to establish.

(b) Completeness of Minimal Logic for Positive Deductive Databases

Although incomplete for full first-order logic, minimal logic is complete for its fragment corresponding to positive – definite or disjunctive – deductive databases without integrity constraints, i.e., if D is a positive, definite or disjunctive, deductive database without integrity constraints, if A is a formula, and if $D \models A$, then $D \vdash_m A$. This result, which is less immediate than the incompleteness of minimal logic for full classical logic, is given in (Bry 1996).

How should this completeness result be interpreted?

Basically, it means that for positive deductive databases without integrity constraints, every answer which can be computed within classical logic can also be computed within minimal logic.

The restriction to positive deductive databases should not be seen as a real restriction, for negation anyway is neither to be interpreted in classical nor minimal logic. Thus, it would not make much sense to build upon a

more general result which would depend upon a interpretation of negation within classical logic. The exclusion of integrity constraints as well is no real restriction. It merely conveys that answers are to be computable – if not computed – from the constructive part of the information system only, as postulated in Section 2.

(c) Extension to More General Information Systems

A generalization of this result to information systems more general than positive, definite or disjunctive deductive databases requires to consider three points. First, how negation is to be interpreted has to be addressed. In (Bry 1996) a semantics is proposed which, for other reasons, also rely on a local notion of inconsistency. Second, the constructive specifications could be generalized from rules to more general constructs allowing for example to express existence assertions such as e.g. null values.

The fact that minimal logic is sufficient a basis for positive – definite or disjunctive – deductive databases makes it reasonable to consider minimal logic instead of full classical first-order logic as a foundation for general information systems. Arguably, it makes sense to keep with not calling for proofs based on double negation elimination or refutation while querying more general information systems than deductive databases.

6 QUERYING INCONSISTENT INFORMATION SYSTEMS

In Section 2.2. it was argued that information systems violating some of their integrity constraints might be queried. In this section, an approach to query answering is proposed that makes it possible to recognize whether an answer to a query has been derived from possibly corrupted data. This approach exploits the local notion of inconsistency argued for in Section 2.3 and formalized in Section 5.

Basically, data giving rise to derive an integrity constraint violation, i.e. according to the formalization of Section 3, data giving rise to a derivation of $\bot$, should be considered as possibly corrupted.

Consider for example a relational database with constructive part $\{p(a), q(a), r(a), r(b)\}$ and integrity constraint $C = p(x) \wedge q(x) \wedge r(x) \rightarrow \bot$. Without further knowledge of the application this database represents, it cannot be recognized which data are erroneous, e.g. whether the violation results from an incorrect argument a in $p(a)$, or in the mere presence of a p-tuple, or from any other reason. The only available knowledge is that $\bot$ is derivable, showing that the constructive part of the database is incorrect.

Intuitively, the answers $p(a)$ and $r(b)$ should not be given the same status. Indeed, $p(a)$ contributes in the violation of an integrity constraint – even though $p(a)$ might well be correct. In contrast, $r(b)$ does not contribute in any integrity constraint violation. If inconsistencies are to be considered local, the

answer $r(b)$ should not be affected by the presence of the (unrelated) inconsistency resulting from $p(a), q(a)$, and $r(a)$. In the following, it is proposed to distinguish between "consistent" and "inconsistent answers". In the example considered here, $r(b)$ is a consistent answer while $p(a)$ is an inconsistent one.

6.1 Consistent and Inconsistent Answers

Consider an information system *IS*. Let *Cons* denote its constructive part, *Norm* its normative part, thus $IS = Cons \cup Norm$. Recall that, according to the formalization of Section 2, *Norm* consists of negated closed formulas, i.e. of expressions of the form $A \rightarrow \bot$ where A is a closed formula.

An *inconsistency kernel* of *IS* is defined as a minimal (for $\subseteq$) subset I of *Cons* such that $I \cup Norm \vdash_m \bot$ (where, as before, $\vdash_m$ denotes derivability in minimal logic).

A closed formula F is said to be an *inconsistent answer* (or *inconsistent consequence*) of the information system *IS* if:

1. F is derivable in minimal logic from the constructive part *Cons* of *IS*, i.e. $Cons \vdash_m F$, and

2. For every $S \subseteq Cons$ such that $S \vdash_m F$, there exists an inconsistency kernel K of *IS* with $S \cap K \neq \emptyset$.

Informally, an inconsistent answer is an answer which cannot be established without making use of some data involved in a derivation of $\bot$, i.e. in an integrity constraint violation.

A closed formula F is said to be a *consistent answer* (or *consistent consequence*) of the information system *IS* if there exists $S \subseteq Cons$ such that:

1. F is derivable in minimal logic from S , i.e. $S \vdash_m F$, and

2. for every inconsistency kernel K of *IS*, $S \cap K = \emptyset$.

Informally, consistent answers are answers that can be computed without using some data involved in a derivation of $\bot$, i.e. in an integrity constraint violation.

It is worth noting that, although referring to derivations of answers only from the constructive part *Cons* of the information system, the definitions above do not preclude to make use of integrity constraints during query evaluation. How this could be done in the context defined here is however an open question.

6.2 Nonmonotonicity of Consistent Answers

Interestingly, the notion of consistent answer introduced in the previous section is nonmonotonic. Let denote by $\vdash_{cons}$ this notion, i.e. $IS \vdash_{cons} A$ denotes that A is a consistent answer (or consequence) of an information system IS.

According to the formalization introduced in Section 2, $\neg A = A \rightarrow \bot$ is a possible integrity constraint. If $IS \vdash_{cons} A$ then, $IS \cup \{\neg A\} \not\vdash_{cons} A$, i.e. $\vdash_{cons}$ is nonmonotonic.

Indeed, if $IS \vdash_{cons} A$ then, by definition, there exists a minimal logic derivation $\mathcal{D}^A$ of A in *Cons*, the constructive part of IS. $\mathcal{D}^A$ can be extended in $Cons \cup \{\neg A\}$ into a minimal logic derivation of $\bot$ as follows:

$$\dfrac{\begin{array}{cc}\mathcal{D}^A & \\ A & A \rightarrow \bot\end{array}}{\bot} \rightarrow \mathrm{E}$$

Thus, the formulas of *Cons* involved in $\mathcal{D}^A$ are in an inconsistency kernel of $IS \cup \{\neg A\}$.

Arguably, the nonmonotonicity of consistent answers can be seen as a further indication that this notion of answer conveys a common, natural intuition of query answering in information systems with integrity constraints.

7 CONCLUSION AND PERSPECTIVES

In this paper, an approach is proposed for answering queries in information systems that might be inconsistent in the sense that they might violate some integrity constraints. The proposed approach consists in distinguishing between two kinds of answers, "consistent answers" that do not depend on data involved in an integrity constraint violation, and "inconsistent answers", that are derived from data violating some integrity constraints. The proposed approach is motivated by practical considerations: Often, queries have to be evaluated in inconsistent information systems.

The proposed approach is quite simple. Formally, it relies on a local notion of inconsistency, which departs from global inconsistency in classical first-order logic. The proposed approach also relies on standard query answering methods. In order to justify the use of these methods, evidence was given that they can be expressed within minimal logic, a constructive restriction of classical first-order logic. It was shown that minimal logic suffices as a foundation of query answering in positive, definite or disjunctive, deductive databases.

This result makes it possible to rely on standard query answering for computing "consistent" and "inconsistent answers". Indeed, if these methods could not be expressed without referring to refutation proofs, then they would be

incompatible with the local consistency notion, from which consistent and inconsistent answers are defined.

Several directions for further research can be indicated.

First, how existing query answering procedures are to be adapted to compute "consistent" and "inconsistent answers" has to be investigated. A reasonable approach could consist in marking the data involved in integrity constraints violations while integrity constraints are checked. This could be done in a manner similar to the technique used in Truth Maintenance Systems (Doyle 1979, de Kleer 1986 a, de Kleer 1986 b, de Kleer 1986 c). However, instead of labeling derived data by the tuples they depend upon like done in these systems, the data items involved in the derivation of $\perp$, i.e. in integrity constraint violations, could be labeled.

Second, it would be interesting to investigate whether and how known semantic query answering methods, i.e. methods that derive answers from (satisfied) integrity constraints, could be adapted to cases where some integrity constraints are violated.

Finally, it would be interesting to investigate extensions of the proposed approach to dynamic integrity constraints.

REFERENCES

Bry, F. (1996) A Compositional Semantics for Logic Programs and Deductive Databases, in *Proc. of the Joint Int. Conf. and Symp. on Logic Programming* MIT Press, 453–467.

Bry, F., Decker, H., and Manthey, R. (1988) A Uniform Approach to Constraint Satisfaction and Constraint Satisfiability in Deductive Databases, in *Proc. Int. Conf. Extending Data Base Technology* Springer-Verlag LNCS 303, 488–505.

Chakravarthy, U.S., Grant, J., and Minker, J. (1990) Logic-Based Approach to Semantic Query Optimization. *ACM Transactions on Database Systems*, **15, 2**, 162–207.

Cholvy, L. (1990) Answering Queries Addressed to a Rule Base. *Revue d'Intelligence Artificielle*, **1, 1**, 79–98.

Clark, K. (1978) Negation as Failure. in (Gallaire and Minker 1978).

Dahl, V. and Saint-Dizier, P. eds. (1988) *Natural Language Understanding and Logic Programming.* North Holland.

Decker, H. (1986) Integrity Enforcement on Deductive Databases. in *Proc. 1st Int. Conf. on Expert Database Systems.*

de Kleer, J. (1986 a) An Assumption-Based TMS. *Artificial Intelligence*, **28**, 127–162.

de Kleer, J. (1986 b) Extending the ATMS. *Artificial Intelligence*, **28**, 163–196.

de Kleer, J. (1986 c) Problem Solving with the ATMS. *Artificial Intelligence*, **28**, 197–224.

Doyle, J. (1979) A Truth Maintenance System. *Artificial Intelligence*, **12**, 231–272.

Fitting, M. (1996) *First-Order Logic and Automated Theorem Proving.* Springer-Verlag. Second Edition 1996.

Gal, A. and Minker, J. (1988) Informative and Cooperative Answers in Databases Using Integrity, in (Dahl and Saint-Dizier 1988).

Gallaire, H. and Minker, J. (1978) *Logic and Data Bases.* Plenum Press, New York.

Gallaire, H. Minker, J. and Nicolas, J.-M. (1984) Logic and Databases: A Deductive Approach. *ACM Computing Surveys*, **16, 2**.

Gentzen, G. (1934) Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, **39**, 176–210.

Green, C. (1969) Theorem Proving by Resolution as a Basis for Question-Answering Systems, in (Meltzer and Michie 1969), 183–205.

Lloyd, J.W. Sonenberg, E.A. and Topor, R.W. (1987) Integrity Constraint Checking in Stratified Databases. *Jour. of Logic Programming*, **4, 4**.

Lobo, J. Minker, J. and Rajasekar, A. (1992) *Foundations of Disjunctive Logic Programming.* MIT Press.

Meltzer, B. and Michie, D. eds. (1969) *Machine Intelligence 4.* Elsevier North-Holland.

Nicolas, J.-M. (1982) Logic for Improving Integrity Checking in Relational Databases. *Acta Informatica.* **18, 3**.

Prawitz, D. (1965) *Natural Deduction. A Proof-Theoretical Study.* Almqvist & Wiksell, Stockholm.

Reiter, R. (1965) On Closed World Databases, in (Gallaire and Minker 1978).

8 BIOGRAPHY

François Bry research interests include principles and applications of automated reasoning, especially logic programming and deductive databases. Since 1994, he is a professor at the Computer Science Institute of Munich University. Formerly, he was with the European Computer-Industry Research Centre, Munich, which he joined in 1985 shortly after its starting. In 1983 and 1984, he worked on statistical databases with INRETS, Paris. In 1981, he worked on text processing as a software engineer with Transac-Alcatel, Paris. From 1979 to 1981, he was with the Combinatorics research group of Claude Berge at Pierre et Marie Curie University of Paris, where he received a PhD in Mathematics in 1981. He contributed as an author, organizer, or editor to scientific conferences and publications. Further interests of François Bry include languages and cultures, fine arts, philosophy, and hiking.